Correction Using Loop Constructs - 0

D - 100 Bye

As do-while loop executes at least once, hence none of the code is unreachable in this case.

Java runtime prints 100 to the console, then it checks boolean expression, which is false.

Hence control goes out of do-while block. Java runtime executes 2nd System.out.println statement to print “Bye” on to the console.

Correction Using Loop Constructs - 1

A - 0 1 2

Inside enhanced for loop, System.out.println(arr[i]); is used instead of System.out.println(i);

When loop executes 1st time, i stores the first array element, which is 2 but System.out.println statement prints arr[2] which is 0. Loop executes in this manner and prints 0 1 2 on to the console.

Correction Using Loop Constructs - 2

C - ArrayIndexOutOfBoundsException is thrown at runtime

Inside enhanced for loop, System.out.println(arr[i]); is used instead of System.out.println(i);

When loop executes 1st time, i stores the first array element, which is 3 but System.out.println statement prints arr[3] and this causes java runtime to throw the instance of ArrayIndexOutOfBoundsException.

Correction Using Loop Constructs - 3

B - It will go in an infinite loop

while loop doesn’t have curly bracket over here, so only System.out.println(x) belongs to while loop.

Above syntax can be written as follows:

```
int x = 5;
while (x < 10) {
    System.out.println(x);
}
x++;
```

As x++; is outside loop, hence value of x is always 5 within loop, 5 < 10 is true for all the iterations and hence infinite loop.

Correction Using Loop Constructs - 4

B -

```
for(int i = 0; i < fruits.length; i++)
  for(int j = 0; j < fruits[i].length ; j++)
    System.out.print(fruits[i][j] + " ");
```

C -

```
for(String [] arr : fruits)
  for(String fruit : arr)
    System.out.print(fruit + " ");
```

Easy question on iterating through 2-dimensional array. Starting index should be 0 and not 1. Enhanced for loop syntax is correct.

As for loops contain 1 statement, hence curly brackets can be ignored.

Correction Using Loop Constructs - 5

A - Compilation error

final boolean flag = false; statement makes flag a compile time constant.

Compiler knows the value of flag, which is false at compile time and hence it gives “Unreachable Code” error.

Correction Using Loop Constructs - 6

A - 25

When start is divisible by 2 [2, 4, 6, 8, 10], continue; statement takes the control to boolean expression and hence sum += start; is not executed.

Hence result is the sum of numbers 1,3,5,7,9.

Correction Using Loop Constructs - 7

A - Compilation error

Variable i is declared inside for loop, hence it is not accessible beyond loop’s body.

System.out.println(i); causes compilation error.

Correction Using Loop Constructs - 8

A - Compilation error

for is a keyword and hence can’t be used as a label.

Java labels follow the identifier naming rules and one rule is that we can’t use java keywords as identifier. Hence, Compilation error.

Correction Using Loop Constructs - 9

D - Program compiles and executes successfully but produces no output.

final boolean flag; flag = false; doesn’t make flag a compile time constant.

Compiler doesn’t know flag’s value at compile-time and hence it allows this syntax.

At runtime, as boolean expression of while loop is false, loop doesn’t execute even once and hence no output.

Correction Using Loop Constructs - 10

B - 3

There is nothing inside loop body, hence loop body is blank.

This loop executes 3 times, for i = 0, i = 1 and i = 2. For i = 3, control goes out of the for loop.

Now, as i is declared outside for loop, hence it is accessible outside loop body.

System.out.println(i); prints 3 to the console.

Correction Using Loop Constructs - 11

B - Prints 5 once.

“outer” and “inner” are valid label names.

On execution, control enters main method and creates int variable i.

On encountering do-while loop, control goes inside and initializes variable i to 5.

Then it executes while loop and it’s boolean expression is always true.

System.out.println(i–); prints 5 to the console first, and then decrements the value of i by 1. So, i becomes 4.

Boolean expression of if(i == 4) evaluates to true. break outer; statement executes and takes the control out of do-while loop.

main method ends and program terminates successfully.

So, 5 gets printed only once.

Correction Using Loop Constructs - 12

C - Unreachable code compilation error

Boolean expression of do-while loop uses literal true (compile-time constant), hence Java compiler knows that this loop is an infinte loop.

It also knows that once at runtime Java Control enters an infinite loop, none of the statements after loop block will get executed.

Hence it marks all the codes after infinite loop as Unreachable Code, which results in compilation error.

If boolean variable was used instead of boolean literal, then this program would have compiled and executed successfully.

```java
public class DoWhileTest1 {
    public static void main(String[] args) {
        boolean flag = true;
        do {
            System.out.println(100);
        } while (flag);

        System.out.println("Bye");
    }
}
```

Above program prints 100 in infinite loop and “Bye” never gets printed.

Correction Using Loop Constructs - 13

B - % $$

Variable ‘arr’ refers to a two-dimensional array. for-each loops are used to iterate the given array.

In 1st iteration of outer loop, str refers to one-dimensional String array {“%”, “$$”}.

In 1st iteration of inner loop, s refers to “%” and “%” will be printed on to the console. Boolean expression of Line n1 evaluates to false so Line n2 is not executed.

In 2nd iteration of inner loop, s refers to “

$$" and "$$

” will be printed on to the console. Boolean expression of Line n1 evaluates to false so Line n2 is not executed.

Iteration of inner for-each loop is over and control executes Line n3. break; statement at Line n3 terminates the outer loop and program ends successfully.

So, output is:

% $$

Correction Using Loop Constructs - 14

G - 12

Body of do-while loop is executed first and then condition is checked for the next iteration.

Initially, flag = false;

1st iteration: Boolean expression of if-block `flag = !flag` = `flag = !false` = `flag = true`: it assigns true to variable ‘flag’ and evaluates to true as well. Line n2 is executed and 1 is printed on to the console. Line n3 takes the control to the boolean expression of Line n5.

2nd iteration: As flag is true, boolean expression at Line n5 evaluates to true and control enters the loop’s body. Boolean expression of if-block `flag = !flag` = `flag = !true` = `flag = false`: it assigns false to variable ‘flag’ and evaluates to false as well. Line n2 and Line n3 are not executed. Line n4 is executed, which prints 2 on to the console. Control goes to the boolean expression of Line n5.

3rd iteration: As flag is false, boolean expression at Line n5 evaluates to false and control exits the loop.

Program terminates successfully after printing 12 on to the console.

Correction Using Loop Constructs - 15

A - 15

No syntax error in the given code.

Initially, i = 1, j = 5 and k = 0.

1st iteration of A: i = 2.

```
1st iteration of B: j = 4.

    1st iteration of C: k = k + i + j = 0 + 2 + 4 = 6. `i == j`
evaluates to false and `i > j` also evaluates to false, hence else
block gets executed. `continue B` takes the control to the loop B.

2nd iteration of B: j = 3.

    1st iteration of C: k = k + i + j = 6 + 2 + 3 = 11. `i == j`
evaluates to false and `i > j` also evaluates to false, hence else
block gets executed. `continue B` takes the control to the loop B.

3rd iteration of B: j = 2.

    1st iteration of C: k = k + i + j = 11 + 2 + 2 = 15. `i == j`
evaluates to true, control breaks out of the loop A.
```

`System.out.println(k);` prints 15 on to the console.

Correction Using Loop Constructs - 16

C - 011

Basic/Regular for loop has following form:

for ([ForInit] ; [Expression] ; [ForUpdate]) {...}

[ForInit] can be local variable initialization or the following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

[ForUpdate] can be following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

The [Expression] must have type boolean or Boolean, or a compile-time error occurs. If [Expression] is left blank, it evaluates to true.

All the expressions can be left blank; for(;;) is a valid for loop and it is an infinite loop as [Expression] is blank and evaluates to true.

In the given code, for [ForInit] and [ForUpdate], `System.out.print(i++);` is used, which is a method invocation statement and hence a valid statement. Given code compiles fine

Let’s check the iterations:

1st iteration: [ForInit] expression is executed, 0 is printed on to the console. i = 1. i < 2 evaluates to true, control goes inside the loop's body and execute `System.out.print(i);` statement. 1 is printed on to the console.

2nd iteration: [ForUpdate] expression is executed, 1 is printed on to the console. i = 2. 2 < 2 evaluates to false, control goes out of the for loop. main method ends and program terminates successfully after printing 011 on to the console.

Correction Using Loop Constructs - 17

D - 102

For the 1st loop variable ‘i’ infers to int type, so no issues for 1st loop and for the 2nd loop variable ‘j’ infers to int type, so no issues for 2nd loop as well.

Let’s check the iteration:

1st iteration of loop one: i = 0

```
1st iteration of loop two: j = 0

    1st iteration of loop three: ctr = 101. As `i == j` evaluates to
true, hence `break two;` gets executed, which takes the control out
of loop two and hence to the increment expression (i++) of loop one.
```

2nd iteration of loop one; i = 1

```
1st iteration of loop two: j = 0

    1st iteration of loop three; ctr = 102. As `i > j` evaluates to
true, hence `break one;` gets executed, which takes the control out
of the loop one.
```

`System.out.println(ctr);` prints 102 on to the console.

Correction Using Loop Constructs - 18

A - Compilation error at Line n3

Line n1 and Line n2 don’t cause any compilation error.

if-else block uses break; and continue; statements. break; will exit the loop and will take the control to Line n4 on the other hand continue; will take the control to Line n2. In both the cases Line n3 will never be executed.

As Compiler knows about it, hence it tags Line n3 as unreachable, which causes compilation error.

Correction Using Loop Constructs - 19

C - Program terminates successfully after printing 1231 on to the console

Basic/Regular for loop has following form:

for ([ForInit] ; [Expression] ; [ForUpdate]) {...}

[ForInit] can be local variable initialization or the following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

[ForUpdate] can be following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

The [Expression] must have type boolean or Boolean, or a compile-time error occurs. If [Expression] is left blank, it evaluates to true.

All the expressions can be left blank; for(;;) is a valid for loop and it is an infinite loop as [Expression] is blank and evaluates to true.

In the given code, for both the loops, `System.out.print(...)` is used as [ForUpdate] expression, which is a MethodInvocation expression and hence a valid statement.

Given code compiles successfully.

Let’s check the iterations:

1st iteration of outer: i = 0. i < 3 evaluates to true.

```
i = 1.

1st iteration of inner: j = 0. j < 3 evaluates to true as j = 0.
Boolean expression `i > ++j` = `1 > 1` evaluates to false. j = 1.

2nd iteration of inner: `System.out.print(j)` prints 1 to the
console. j < 3 evaluates to true as j = 1. Boolean expression `i >
++j` = `1 > 2` evaluates to false. j = 2.

3rd iteration of inner: `System.out.print(j)` prints 2 to the
console. j < 3 evaluates to true as j = 2. Boolean expression `i >
++j` = `1 > 3` evaluates to false. j = 3.

4th iteration of inner: `System.out.print(j)` prints 3 to the
console. j < 3 evaluates to false as j = 3. Control goes out of
inner for loop and to the [ForUpdate] expression of outer loop.
```

2nd iteration of outer: `System.out.print(i)` prints 1 to the console. i < 3 evaluates to true as i = 1.

```
i = 2.

1st iteration of inner: j = 0. j < 3 evaluates to true as j = 0.
Boolean expression `i > ++j` = `2 > 1` evaluates to true. j = 1. `
break outer;` takes the control out of the outer for loop.
```

Program terminates successfully after printing 1231 on to the console.

Correction Using Loop Constructs - 20

B - 33

Basic/Regular for loop has following form:

for ([ForInit] ; [Expression] ; [ForUpdate]) {...}

[ForInit] can be local variable initialization or the following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

[ForUpdate] can be following expressions:

Assignment

PreIncrementExpression

PreDecrementExpression

PostIncrementExpression

PostDecrementExpression

MethodInvocation

ClassInstanceCreationExpression

The [Expression] must have type boolean or Boolean, or a compile-time error occurs. If [Expression] is left blank, it evaluates to true.

All the expressions can be left blank; for(;;) is a valid for loop and it is an infinite loop as [Expression] is blank and evaluates to true.

Multiple comma separated statements are allowed for [ForInit] and [ForUpdate] expressions, where as [Expression] must be single expression which results in boolean or Boolean.

In the given for loop:

[ForInit] = int x = 10, y = 11, z = 12: It is allowed. 3 variables are declared and initialized. x = 10, y = 11 & z = 12.

[Expression] = y > x && z > y = (y > x) && (z > y) [Relational operator has higher precedence than logical AND]. This expression is valid and results in boolean value.

[ForUpdate] = y++, z -= 2. It is allowed. y is incremented by 1 and z is decremented by 2.

Let's check the loop's iteration:

1st iteration: x = 10, y = 11, z = 12. (y > x) && (z > y) = (11 > 10) && (12 > 11) = true && true = true. Loop's body is executed and prints x + y + z = 10 + 11 + 12 = 33 on to the console.

2nd iteration: [ForUpdate] is executed. y = 12, z = 10. (y > x) && (z > y) = (12 > 10) && (10 > 12) = true && false = false.

Control goes out of for loop and program terminates successfully.

Loop's body executes once and prints 33 on to the console.